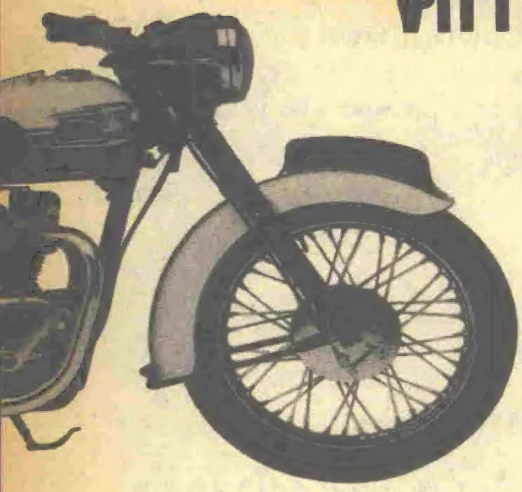
ศึกษาภาษาสมอลทอล์ค : สกุลกับกิจกรรม

บุญเลิศ เอี่ยมทัศนา



ท่านที่เคยคุ้นเคยกับภาษาปาสคาลมาก่อนอาจสงสัยว่าในภาษาสมอลทอล์คนั้น ผู้เขียนโปรแกรมสามารถกำหนดโปรแกรมย่อยได้หรือไม่ โดยเฉพาะเมื่อผมกล่าวถึงออปเจ็คกับสื่อความในตอนที่แล้ว จะเห็นว่ามีแต่การใช้สื่อความซึ่งเทียบได้กับการเรียกใช้ฟังก์ชันในภาษาปาสคาลเท่านั้น

ในภาษาสมอลทอล์คให้ผู้เขียนโปรแกรมสามารถกำหนดโปรแกรมย่อยได้เช่นกัน แต่เป็นการทำงานคนละขั้นคนละตอนกัน ไม่ใช่กำหนดแล้วเรียกใช้ในที่แห่งเดียวกันเช่นในภาษาปาสคาล ต้องมาดำเนินการในเรื่องของสกุลกับกิจกรรม

ดังนั้นเพื่อความเข้าใจง่ายขึ้น ผมจึงขอแบ่งทั้งโปรแกรมกับผู้เขียนโปรแกรมออกเป็น 2 ระดับ คือระดับสื่อความดังที่กล่าวในฉบับที่แล้ว และระดับกิจกรรมดังที่จะกล่าวในฉบับนี้

อนึ่งผมขอเรียนให้ทราบก่อนนะครับว่า **กิจกรรมคือโปรแกรมย่อยที่ประกอบด้วยชื่อและตัวโปรแกรม** ซึ่งผู้เขียนโปรแกรมสื่อความจะนำ**ชื่อกิจกรรม**นี้ไปใช้เป็นสื่อความ อีกเรื่องหนึ่งคือกิจกรรมในภาษาสมอลทอล์คจะมีลักษณะเหมือนโปรแกรมย่อยในภาษาซี และเทียบได้กับฟังก์ชันในภาษาปาสคาลคือต้องมีการส่งค่ากลับ 1 ค่าเสมอ

"การศึกษาภาษาสมอลทอล์ค และเขียนโปรแกรมแบบโอไอพี ควรเรียนรู้เกี่ยวกับคุณสมบัติวงศ์สกุล การตั้งสกุลและการใช้คำสั่งกิจกรรมต่าง ๆ ได้ด้วย"

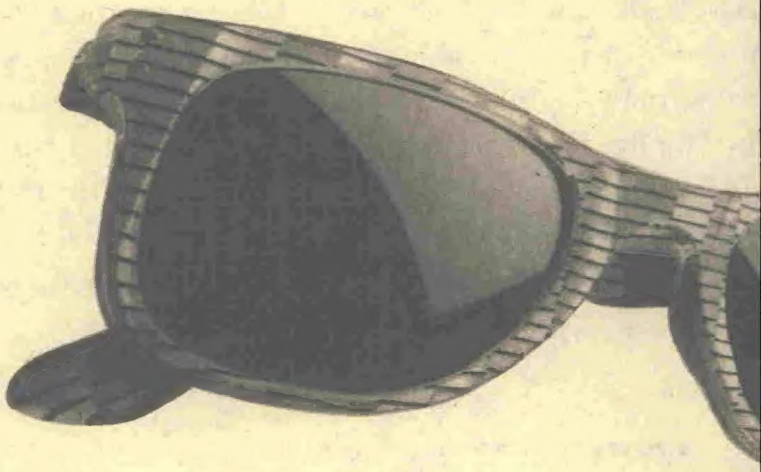
สกุล (Classes)

เช่นเดียวกันกับออปเจ็คกับสื่อความที่แยกจากกันมาอธิบายได้ยาก แต่สกุลกับกิจกรรมยิ่งยากกว่า ที่แยกหัวข้อก็เพื่อจะให้เห็นว่าจะเน้นในเรื่องใดเท่านั้น สำหรับในเรื่องของสกุลมีเรื่องที่จะขอนำมากล่าวดังนี้คือ

การสืบสกุล (Class hierachy)

ขอให้ดูของจริงกันเลยตามรูปที่ 1 แสดงการสืบสกุลใน ST/V ที่พิมพ์ลดหลั่นกันเพื่อแสดงว่า ใครเป็นลูกหลานใคร ต้นตระกูลมีเพียงหนึ่งเดียวคือ Object (อาจดูได้ยากเล็กน้อย) มีลูกเป็น Behavior, BitBit,..., Dispatcher, Dispatch Manager เรื่อยไปถึง Undefined Objects

ซึ่งลูก ๆ ของ Object อาจมีลูกหลานของตนเองอีกเช่น Magnitude



```
Object
    Behavior
        Class
        MetaClass
    BirBlt
        CharacterScanner
        Pen
        Animation
        Commander
    Boolean
        False
        True
    ClassBrowser
    ClassHierarchyBrowser
    ClassReader
    Collection
        Bag
        IndexedCollection
            FixedSizeCollection
                Array
                Bitmap
                ByteArray
                    CommpiledMethod
                    FileControlBlock
                Interval
                String
                    Symbol
            OrderedCollection
                Process
                SortedCollection
        Set
            Dictionary
                IdentityDictionary
                    MethodDictionary
                SystemDictionary
            SymbolSet
    Compiler
    Context
    CursorManager
        NoMouseCursor
    DemoClass
    Directory
    DiskBrowser
    Dispatcher
        GraphDispatcher
        PointDispatcher
        ScreenDispatcher
        ScrollDispatcher
            FormEditor
            ListSelector
            TextEditor
                PromptEditor
            TopDispatcher
    DispatchManager
    DisplayObject
        DisplayMedium
            Form
                DisplayScreen
                SelectorForm
        DisplayString
        InfiniteForm
    File
    Inspector
        Debugger
        DictionaryInspector
    Magnitude
        Association
        Character
        Date
        Number
            Float
            Fraction
            Integer
                LargeNegativeInteger
                LargePositiveInteger
                SmallInteger
        Time
    Menu
    Message
    Pane
        SubPane
            GraphPane
            ListPane
            TextPane
        TopPane
    Pattern
        WildPattern
    Point
    Prompter
    Rectangle
    Stream
        ReadStream
        WriteStream
            ReadWriteStream
                FileStream
                TerminalStream
    StringModel
    TextSelector
    UndefinedObject
    SwappedOutObject
```

ที่ 1 แสดงการสืบสกุลของสกุลต่าง ๆ ในสมอลทอล์ค

ก 5 คน และมีหลานและเหลนอีก
Number เป็นต้น

ความหมายของสกุลคือ Object
นต้นสกุลใหญ่ ลูกของ Object คือ
สกุลรอง พวกหลานและเหลนคือต้น
ลรองรองลงไปอีก แต่ในสมอลทอล์ค
ยกว่าสกุล (class) ทั้งหมด

สำหรับความสัมพันธ์ภายในสกุล
ยบได้กับวงศ์สกุลของมนุษย์เรา และ
ยบได้กับการแตกตัวของโปรแกรม
ยในภาษาปาสคาลได้โดยตรง) คือ
บสกุล เช่น พวกลูก หลาน เหลน
ฯ สามารถใช้สมบัติของบรรพบุรุษ
โดยตรงของตนได้ **สมบัติ**หมายถึงตัว-
แปรและกิจกรรม (ดังจะกล่าวต่อไป)

แต่ในทางตรงข้ามบรรพบุรุษไม่
สามารถใช้สมบัติของผู้สืบสกุลได้ และ
พี่น้องกันจะใช้สมบัติของกันและกัน
ไม่ได้ (รวมทั้งลูกหลานในสกุลนั้น ๆ

ด้วย) แต่มีค่าคงตัวและตัวแปรจำนวน
หนึ่งที่กำหนดไว้เป็นส่วนกลาง (pool)
ให้ใช้ด้วยกัน

สกุลกับสมาชิก

จากสกุลก็มาถึงครอบครัวแต่อาจ
จะแตกต่างกับของจริงคือ เมื่อสกุลใด
มีลูกจะต้องถือว่าลูกได้แยกครอบครัว
ไปอย่างเด็ดขาด ภายในครอบครัวจึงคง
เหลือแต่ตัวแปร (ของใช้) กับกิจกรรม
(คนใช้) ซึ่งทั้งตัวแปรและกิจกรรมนี้
แยกเป็น 2 พวกคือ **ส่วนตัวเพื่อใช้ใน
ครอบครัวเท่านั้น** กับ**ส่วนกลางเพื่อให้
ผู้สืบสกุลนำไปใช้ได้** ซึ่งของที่เป็นส่วน
กลางจะกำหนดไว้ในส่วนหัวของสกุล
ดังจะกล่าวต่อไป

ในการเขียนโปรแกรมสื่อความเรา
มองออปเจ็คในฐานะกรรมของกริยา
คือสื่อความ หรือมองออปเจ็คในฐานะ
ผู้รับปฏิบัติ แต่ในเรื่องของสกุลจะถือ
ว่าออปเจ็คเป็นสมาชิกคือ Instance
(ซึ่งให้ความหมายไว้ว่าเป็น member)

ออปเจ็คทุกตัวจะต้องมีสังกัดคือ
เป็นสมาชิกของสกุลใดสกุลหนึ่ง เช่น
เลขจำนวนเป็นสมาชิกของ Number
สตริงเป็นสมาชิกของ String, เลขเศษ
ส่วน เช่น (๑/๕) เป็นสมาชิกของ Fraction เป็นต้น

ส่วนในด้านการเขียนโปรแกรม
กิจกรรม จะกำหนดตัวแปรไว้รองรับ
ออปเจ็คเรียกว่า Instance Variable
คือตัวแปรสมาชิก มีกิจกรรมที่จะดำเนิน
การกับตัวแปรสมาชิกเรียกว่า Instance
Method คือกิจกรรมสมาชิก ที่นำไป
ใช้เป็นสื่อความ

นอกจากนี้ยังมีตัวแปรที่เป็นโกล-
บอล เพื่อเก็บคำที่ใช้เป็นส่วนกลางของ
สกุลนั้นเรียกว่า Class Variable คือ
ตัวแปรของสกุล และมีกิจกรรมที่จะ
ดำเนินการกับออปเจ็คโดยตรงซึ่งส่วน
มากจะเป็นการเตรียมตัวแปรสมาชิก
เช่น คำว่า new: เป็นต้น เรียกว่า Class
Method คือกิจกรรมของสกุล

```
PATTERN

Inherits From                Object

Inherited By                 WildPattern

Named Instance Variables
  fisst
    Contains the first element in the pattern collection.

Class Variables
  WildcardChar
    Contains a Character which when appears in the pattern
    collection will match zero or more elements.

Pool Dictionaries            ( None )

Class Methods
  initialize
    Initialize the class variable WildcardChar.

Instance Methods
  input
    Answer the pattern string to which the input is compared.
```

รูปที่ 2 เรื่องราวของสกุล Pattern ที่อยู่ในเอ็นไซโคปีเดีย

ตัวอย่าง

ในเอ็นไซโคปีเดียจะให้ข้อมูลของสกุลต่าง ๆ แก่เรา ซึ่งผมขอยกสกุล Pattern มาเป็นตัวอย่างดังในรูปที่ 2 แต่ตัดตอนมาเพียงบางส่วนเท่านั้น มีรายละเอียดต่าง ๆ ดังนี้คือ

- เริ่มด้วยแสดงให้ทราบว่าสกุลนั้น ๆ ทำอะไรได้บ้างเทียบได้กับว่ามีอาชีพอะไร (ไม่มีในรูปที่ 2)

- Inherite From แจ้งว่าสืบสกุลมาจากใคร กรณีนี้คือ Object

- Inherited By มีใครเป็นลูกบ้าง

- Instance Variable คือตัวแปรสมาชิก แยกเป็น 2 แบบคือ Named Instance Variable หมายถึงตัวแปรสมาชิกที่เป็นชื่อ เช่น first ในรูปที่ 2 และ Indexed Instance Variable คือตัวแปรสมาชิกที่เก็บค่าอินเด็กซ์

- Instance Method คือกิจกรรมสมาชิก เช่น input

- Class Variable คือตัวแปรของสกุล เช่น WildcardChar

- Class methods คือกิจกรรมของสกุล เช่น initialize

- Pool Dictionaries คือค่าคงตัวและตัวแปรที่กำหนดไว้ให้ใช้เป็นส่วนรวม

ในการเขียนโปรแกรมเมื่อกำหนดสกุลใหม่ต้องกำหนดสิ่งต่าง ๆ ดังกล่าวมานี้ด้วยเว้นผู้สืบสกุล และก็ควรทำบันทึกตามรูปแบบที่เขียนไว้ในเอนไซโคปีเดียคือตามที่กล่าวมานี้ สำหรับที่กล่าวไว้ว่า "กำหนดสกุลใหม่" นั้น

```
Number subclass: #Fraction
  instanceVariableNames:
    'numerator denominator '
  classVariableNames: ''
  poolDictionaries: ''
```

รูปที่ 3 ส่วนหัวของสกุล Fraction

ความจริงเรากำหนดสกุลใหม่แท้ ไม่ได้ ทำได้แต่เพียงการแทรกสกุลเพรา อย่างน้อยต้องเป็นลูกของ Object ก็ข ให้เป็นที่เข้าใจกัน

กิจกรรม (Methods)

ดังที่กล่าวมาแล้วว่าเป็นการยา ที่จะแยกเรื่องราวของสกุลกับกิจกรรม ออกจากกัน และได้กล่าวถึงเรื่องรา โดยทั่วไปของกิจกรรมไปแล้ว ในหัวข้ นี้จะกล่าวถึงรูปแบบของกิจกรรม (ท ของสกุลและสมาชิก) เพียงเพื่อให้เห็ รูปร่างก่อน ส่วนจะดำเนินการได้อย่าง จะกล่าวในหัวข้อ "การต่อเติมกิจกรรม

ขอยกตัวอย่างสกุล Fraction ซึ่ เป็นสกุลที่จะดำเนินการกับเลขเศษส่ว เช่น (36/5) จะเป็นเลขเศษส่วนในภา สมอลทอล์ค หากให้เก็บค่าจะเก็บเช่น ไม่เก็บเป็น 7.2 มีส่วนซึ่งก็คงต้องม เรียกว่า ส่วนหัวดังรูปที่ 3 เป็นไปตาม กล่าวในหัวข้อ "ตัวอย่าง"

ซึ่งได้กำหนดตัวแปรสมาชิกไว้ ตัวคือ numerator สำหรับเก็บค่าเศ เช่น 36 และ denominator สำหรับเก็ ค่าส่วน เช่น 5

ภายในสกุล Fraction ได้กำหน กิจกรรมไว้มากมาย แต่ไม่มีกิจกรรม จะให้ค่าผลบวกของเศษกับส่วน สมม ว่าเราจะเขียนกิจกรรมนี้และให้ชื่อเป็ sum จะเขียนได้เป็น

```
sum
^(numerator + denominato
```

โดยที่บรรทัดแรกเป็นการกำหน ชื่อกิจกรรม (ฟังก์ชัน) บรรทัดที่ส เป็นตัวโปรแกรมให้ส่งค่าผลบวกข numerator กับ denominator กลับ ในฐานะฟังก์ชัน sum เช่น เมื่อให้โป เมินผลประโยคข้างล่างจะได้คำตอบเป็ 11

(7/4) sum

เพื่อให้เห็นตัวอย่างของโปรแกร ที่ซับซ้อนมากขึ้นขอยกกิจกรรม * (

น binary message) เช่น เมื่อเรา
ยนเป็นประโยคว่า

(2/3) * (5/7)

จะเป็นการคูณในแบบเลขเศษส่วน
มที่ได้กำหนดกิจกรรมไว้แล้วในสกุล
action ดังโปรแกรมในรูปที่ 4 เช่น
ยวกันกับบรรทัดแรกคือชื่อกิจกรรม *
ยอาร์กิวเมนต์คือ aNumber เป็นตัว
ปร บรรทัดที่ 2 และบรรทัดที่ 3 เป็น
โปรแกรม ซึ่งจะมีการดำเนินการ
นี้คือ

- ในแต่ละวงเล็บจะทำกิจกรรม
merator หรือ denominator ก่อน
ามหลักการที่ว่าจะทำ unary message
น binary message) ซึ่งกิจกรรม
ดู่นี้อยู่ใน Fraction ได้ผลเป็นเลข
นวนเต็ม (ไม่ใช่เลขเศษส่วน)
- ดังนั้นการคูณคือกิจกรรม * ใน
สองวงเล็บจะเป็นการคูณเลขจำนวน
ม โดยใช้กิจกรรมนี้ใน Integer
- สำหรับการหารคือกิจกรรม /
ดำเนินการด้วยกิจกรรมนี้ใน Integer
นกัน แต่ถ้าหารไม่ลงตัวจะส่งค่าคืน
เป็นเลขเศษส่วน

อีกเรื่องหนึ่งที่ควรทราบคือใน
ษาสมอลทอล์คได้แยกชื่อตัวแปร, ชื่อ
กรรม และชื่อสกุลออกจากกัน ทำ
สามารถตั้งชื่อซ้ำกันได้ เช่น ใน
action มีกิจกรรมที่จะให้ค่าเศษคืน
ก็ตั้งชื่อเป็น numerator เช่นกัน และ
ยนโปรแกรมได้ดังนี้

```
numerator
    ^numerator
```

โดยที่บรรทัดแรกหมายถึงชื่อกิจ-
รม ส่วนบรรทัดที่สองหมายถึงตัว
ร

elf ตัวแปรพิเศษ

ในเรื่องของกิจกรรมขอแสดงอีก
ัวอย่างครับ เพื่อนำไปเป็นตัวอย่าง
การแทรกกิจกรรมเข้าไปในสกุลด้วย
ที่ผมตั้งหัวข้อเช่นนี้เพื่อเน้นคำว่า self ไว้ให้สะดุดตา เพราะจะแสดงการใช้ในตัวอย่างนี้

โจทย์ของตัวอย่างนี้คือ ในสกุล Fraction ยังไม่มีกิจกรรมที่จะส่งค่าเศษส่วนที่เป็นเศษของการหารแล้วคืนมา จะเขียนโปรแกรมได้ดังรูปที่ 5

บรรทัดแรกเป็นกิจกรรมซึ่งตั้งชื่อซ้ำกับสกุล (เสียดื้อ ๆ เท่านั้นเอง) บรรทัดที่สองเป็นคำอธิบายอยู่ในเครื่องหมายคำพูดคู่ ดังที่ในภาษาปาสคาลใช้เครื่องหมาย { } บรรทัดที่ 3 เป็นตัวโปรแกรม เขียนได้ง่ายมากครับ

โดยที่ self ซึ่งน่าจะหมายถึงตัวเองคือ fraction แต่กลับหมายถึงออปเจ็ค (เรื่องนี้ต้องจำไว้ด้วยครับ) เช่น หากเราเขียนประโยคว่า (5/3) fraction ค่า (5/3) คือออปเจ็ค และคือ self

ความหมายของโปรแกรมคือ ให้ลบตนเอง (5/3) ด้วยค่าที่เป็นผลของการหารตนเองในแบบปัดเศษทิ้ง (1) แล้วส่งค่า (2/3) คืนไป

การดำเนินการกับวงศ์สกุล (The Class Hierachy Browser)

จนถึงจุดนี้คุณผู้อ่านอาจสงสัยว่าเรื่องของวงศ์สกุลก็รู้แล้ว ตัวแปรก็รู้แล้ว กิจกรรมก็รู้แล้ว แต่ยังไม่เห็นทำอะไรได้เลย ผมไม่ทราบว่าเป็นเรื่องดีหรือไม่คือการเขียนโปรแกรมในภาษาสมอลทอล์คนั้น เมื่อเขียนเสร็จแล้วจะต้องแทรกเข้าไปในระบบของสมอลทอล์ค แต่ยังดีครับที่สามารถลบทิ้งได้โดยง่าย ไม่เหมือนภาษา FORTH ที่เหมือนกับการงัดทิ้ง

ในการนี้ ST/V ได้เตรียมสิ่งอำนวยความสะดวกไว้ให้เราแล้วคือ Class Hierarchy Browser ซึ่งให้เราสามารถทำได้ทั้งการแทรกคือเพิ่ม, การถอดถอนคือลบ และการแก้คือแก้ กับตัวแปร, กิจกรรม และตัวสกุลเอง ซึ่งจะขอรวมเรียกว่าการแทรกลบแก้ ทำได้ทั้งแบบง่ายและแบบยาก จะกล่าวถึงวิธีง่าย ๆ ก่อนครับ ส่วนวิธียากหรือวิธีดำเนินการโดยตรงจะขอนำไปกล่าวในแต่ละเรื่อง

วิธีแรกที่ผมว่าง่ายคือ ดำเนินการผ่านเมนู มีขั้นตอนดำเนินการดังนี้

* เรียกใช้เมนูหลักเลือกรายการ browser class แล้วกำหนดหน้าต่าง

```
* aNumber
        "Answer the result of multiplying
         the receiver by aNumber."
    ^(numerator * aNumber numerator) /
       (denominator * aNumber denominator)
```

รูปที่ 4 โปรแกรมของกิจกรรม * (คูณ)

```
fraction
    "Answer the reciever minus its integral part."
    ^self - self truncated
```

รูปที่ 5 โปรแกรมของกิจกรรม Fraction

(ดูในไมโครฯ #56 ครับ) เครื่องจะแสดงหน้าต่างดังรูปที่ 6 มีบานหน้าต่างอยู่ 4 บานคือ

- **บานสกุล** อยู่ด้านบนซ้ายแสดงชื่อสกุลต่าง ๆ เริ่มตั้งแต่ Object หากต้องการเลื่อนหน้าต่างลงให้กดคีย์ PgDn (ขึ้น PgUp) ถ้าต้องการเลือกสกุลใดให้เลื่อนเคอร์เซอร์ไปที่สกุลนั้นแล้วกดคีย์เลือก (+) แต่จะเห็นว่าบางสกุลมีเครื่องหมายจุด จุดต่อท้ายชื่อแสดงว่ามีผู้สืบสกุล หากต้องการดูหรือเลือกสกุลรองของสกุลใดให้เลือกสกุลนั้น แล้วเรียกเมนูรองของบานหน้าต่างนี้ (กดคีย์ Del) เลือกรายการ Hide/show (หน้าต่างเมนูจะหายไป) แสดงสกุลรองของสกุลที่เลือก

- **บานกิจกรรม** อยู่ด้านบนขวาตามรูปที่ 6 ยังว่างอยู่ หากเลือกสกุลใดจะแสดงกิจกรรมของสกุลนั้นในบานนี้ให้ดูหรือเลือกต่อไป

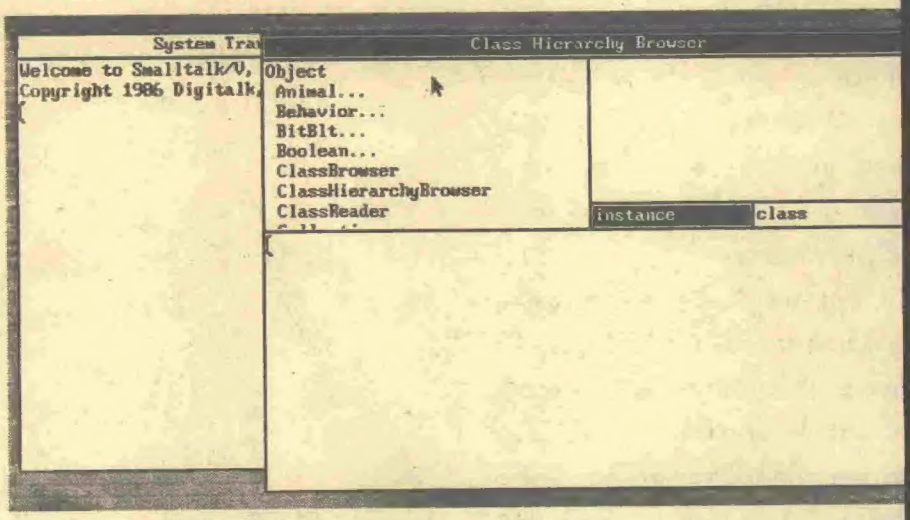
- **บานอีดิต** อยู่ด้านล่างตามรูปที่ 6 ยังว่างอยู่ หากเลือกกิจกรรมใดจะแสดงโปรแกรมของกิจกรรมนั้นในบานนี้ให้ดำเนินการอีดิตต่อไปดังในรูปที่ 7 ตามรูปนี้เป็นการเลือกสกุล Fraction และกิจกรรม / คือการหารเลขเศษส่วน

- **บานเลือกสมาชิกหรือสกุล** ระหว่างบานอีดิตกับบานกิจกรรม จะเห็นว่ามีหน้าต่างบานเล็ก ๆ บรรทัดเดียวอยู่บานหนึ่ง มีข้อความว่า instance กับ class ตามรูปที่กล่าวมาเป็น instance คือดำเนินการกับกิจกรรมสมาชิก หากต้องการดำเนินการกับกิจกรรมที่เป็นของสกุลให้เลือก class

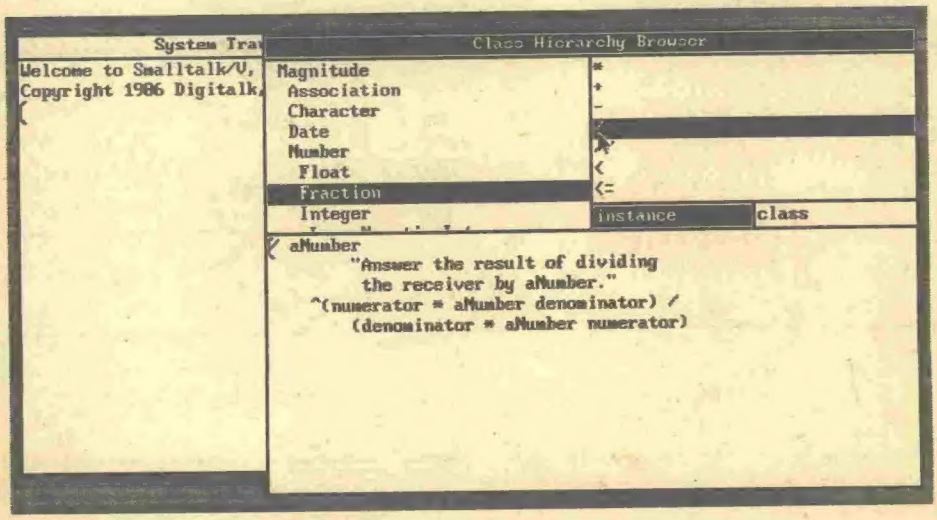
* ดำเนินการแทรกลบแก้ เป็นเรื่ ละเอียดในแต่ละเรื่อง จึงจะขอนำ กล่าวในแต่ละเรื่องต่อไป ตอนนี้ขอ ทดลองเลือกสกุล เลือกกิจกรรม ซึ่ง กลับไปกลับมาอย่างไรก็ได้ไปก่อนคร เพื่อทำความคุ้นเคยกับรูปแบบของโป แกรมในภาษานี้

อนึ่งเมื่อคุณเลือกสกุลและก่อน จะเลือกกิจกรรม เครื่องจะแสดงส่วนห ของสกุลดังของ Fraction ในรูปที่ ที่บานอีดิต

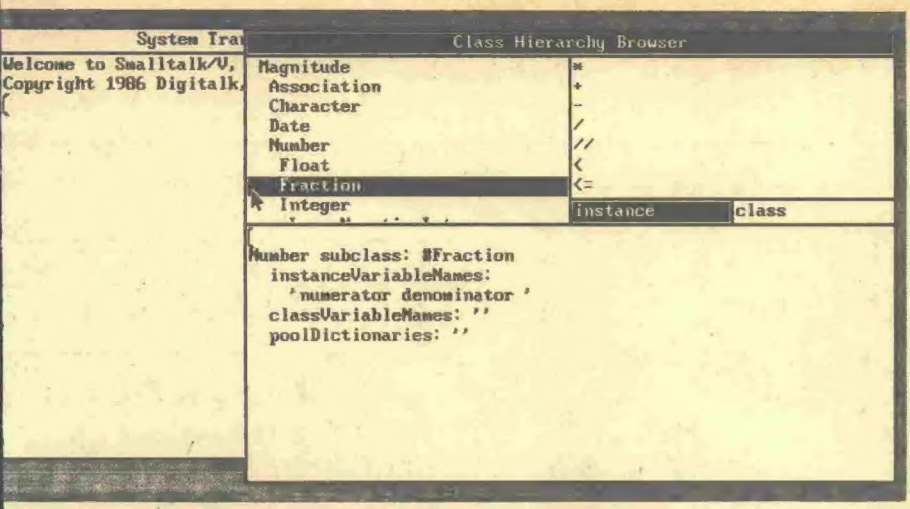
สำหรับท่านที่ติดตามบทความข ผมด้วยการทดลองใช้ ST/V ตามผ ผมคิดว่าตอนนี้ควรจะออกจาก ST/ ก่อน เพราะการดำเนินการต่อไปจะเก็ กิจกรรมลงระบบ ST/V ซึ่งมันจะเก็ ทุกอย่างรวมทั้งตัวแปรโกลบอลด้วย



รูปที่ 6 แสดงหน้าต่างสกุล (คือหน้าต่างอยู่ทางด้านขวาของจอภาพ) เมื่อเริ่มเปิดหน้าต่างนี้



รูปที่ 7 แสดงหน้าต่างสกุล เมื่อเลือ สกุล Fraction และกิจกรรม ข้อความในบานอีดิตคือโปร แกรมของกิจกรรมนี้



ปที่ 8 แสดงภาพเมื่อเลือกสกุล Fraction

ารแทรกกิจกรรม

หลังจากที่เราเขียนโปรแกรมของ กรรมแล้วดังเช่นของ fraction (รูป 5) ขั้นตอนต่อไปคือการแทรกกิจ- รมนี้กับสกุลที่ต้องการ ในกรณีนี้ Fraction (ชื่อเดียวกัน) ซึ่งมีวิธีให้ อก 2 วิธีคือ ดำเนินการผ่านเมนูหรือ เนินการโดยตรง

การดำเนินการโดยตรงทำได้โดย ะเมินผลประโยคในรูปที่ 9 ด้วย it ซึ่งจะให้กำหนดหน้าต่างและมี การต่าง ๆ เช่นเดียวกันกับดำเนิน รผ่านเมนู แตกต่างกันตรงที่ในหน้า- งบานสกุลจะแสดงรายการสกุลเฉพาะ ำหนดในคำสั่งเท่านั้น

ขั้นตอนของการแทรกกิจกรรมให้ดำเนินการ ดังนี้ครับ

- เลือกสกุลที่ต้องการคือ Fraction
- เลื่อนเคอร์เซอร์ไปบานกิจกรรม เรียกใช้เมนูรองเลือกรายการ new method
- ST/V จะแสดงรูปแบบของกิจกรรมให้ดูเป็นตัวอย่างในบานอีดิต ดังในรูปที่ 10
- เลื่อนเคอร์เซอร์ไปที่บานอีดิต แล้วป้อนโปรแกรมที่เตรียมไว้คือตามรูปที่ 5
- เรียกเมนูรองของอีดิตเลือกราย- การ save เท่านี้เองครับ

สำหรับการแทรกกิจกรรมของสกุลคงดำเนินการเช่นเดียวกัน โดยเลือก class ในหน้าต่างเล็กเมื่อเริ่มต้น

```
ClassHierarchyBrowser new openOn: (Array with: Integer
  with: Fraction with: String with: DemoClass)
```

ที่ 9 ประโยคที่ใช้ในการดำเนินการกับสกุลโดยตรง

```
messagePattern
        "comment"
    | temporaries |
    statements
```

ที่ 10 รูปแบบของโปรแกรมกิจกรรม

การแทรกตัวแปร

การแทรกตัวแปรไม่ว่าจะเป็นของสมาชิกหรือของสกุลทำได้โดย เลือกสกุลที่ต้องการจะแทรกตัวแปร (หน้าต่างบนซ้าย) เพื่อเป็นตัวอย่างขอให้เลือกสกุล DemoClass จะปรากฏรูปแบบของสกุลนี้ที่บานอีดิต ให้เลื่อนเคอร์เซอร์ไปที่ท้ายคำว่า ClassVariableName: (ซึ่งเดิมตามด้วยสตริงว่าง) แล้วป้อนชื่อตัวแปรที่ต้องการคือ MandalaCount ให้เป็นดังในรูปที่ 11

เป็นการต่อเติมตัวแปรของสกุลชื่อ MandalaCount ส่วนตัวแปรสมาชิกของเดิมมีอยู่แล้ว 2 ตัวคือ pen กับ rectangle แต่การต่อเติมตัวแปรไม่ใช่ว่าจะต่อเติมโดยปราศจากเหตุผล ตามตัวอย่างนี้ก็เพื่อแสดงการแก้กิจกรรมครับ

การแก้กิจกรรม

ในบางโอกาสเราอาจจำเป็นต้องปรับปรุงกิจกรรมเดิม (แต่โอกาสที่จะใช้จริงคือกับสกุลที่เรากำหนดขึ้นใหม่) เช่น สมมติว่าเราต้องการทราบว่ามีการเรียกใช้กิจกรรมใดมากน้อยเท่าไร สิ่งแรกที่ต้องคำนึงถึง คือ จะต้องมีตัวแปรเก็บจำนวนครั้งไว้ ซึ่งตัวแปรนี้ไม่เกี่ยวข้องกับออปเจ็คของกิจกรรมใด จึงต้องให้เป็นตัวแปรของสกุล เพราะต้องการให้เก็บค่าไว้จึงต้องเป็นตัวแปรแบบโกล

```
Object subclass: #DemoClass
  instanceVariableNames: 'pen rectangle '
 classVariableNames: 'MandalaCount '
 poolDictionaries: ''
```

รูปที่ 11 แสดงการแทรกตัวแปร

บอลดังที่ให้แทรกไว้แล้วคือ Mandala-Count

ทางด้านโปรแกรมก็จะต้องมีโปรแกรมเพิ่มค่านี้ทุกครั้งที่มีการใช้กิจกรรมนั้นดังในรูปที่ 12 สำหรับกิจกรรมได้เลือก Mandala นั่นคือ เราต้องนำส่วนของโปรแกรมตามรูปที่ 12 ไปแทรกไว้ในโปรแกรมของกิจกรรม Mandala

ทำได้โดยเลือกกิจกรรม Mandala ในสกุล DemoClass แล้วเลื่อนเคอร์เซอร์ไปที่ท้ายโปรแกรมเดิมเติม . (จุด) หากของเดิมไม่มี ขึ้นบรรทัดใหม่ป้อนโปรแกรมใหม่นี้ต่อท้าย

ยังไม่จบครับ เพราะในภาษาสมอลทอล์คไม่ให้ติดต่อกับตัวแปรในสกุลโดยตรง ต้องเขียนกิจกรรมใหม่เพื่ออ่านค่านี้ดังในรูปที่ 13 ขอให้แทรกกิจกรรมนี้ในสกุล DemoClass ตามวิธีที่กล่าวมาแล้วด้วย

เพื่อเป็นการทดลองใช้งานขอให้ทดสอบประโยคในรูปที่ 14 ด้วย show it ก่อนและหลังการรันโปรแกรมที่ใช้ Mandala ใน run demo (อยู่ในเมนูหลัก)

งานของผู้เขียนโปรแกรมกิจกรรม

ความจริงเรื่องราวต่าง ๆ ที่ผมกล่าวมาในฉบับนี้เป็นงานของผู้เขียนโปรแกรมกิจกรรมทั้งนั้น แต่เป็นส่วนปลีกย่อย ในหัวข้อนี้ผมจึงจะกล่าวถึงเรื่องใหญ่ (แต่โปรแกรมไม่จำเป็นต้องใหญ่ตามไปด้วยครับ) คือ

การกำหนดสกุลใหม่

แต่ก่อนอื่นขอให้ยกเลิกหน้าต่างที่เปิดเพื่อใช้ดำเนินการกับสกุลตามที่กล่าวมาแล้วก่อนครับ เพราะเป็นการใช้สิ่งอำนวยความสะดวกนี้โดยตรง ระบุสกุลไว้เพียง 4 สกุลเท่านั้น เมื่อปิดแล้วขอให้เปิดใหม่โดยดำเนินการผ่านเมนู ซึ่งจะแสดงชื่อสกุลทุกสกุลที่บาน

```
MandalaCount isNil
    ifTrue:  [MandalaCount := 1]
    ifFalse: [MandalaCount := MandalaCount + 1]
```

รูปที่ 12 ประโยคที่เพิ่มค่าตัวแปร

```
mandalaCount
    ^MandalaCount
```

รูปที่ 13 กิจกรรมให้ค่าในตัวแปร MandalaCount

```
DemoClass new mandalaCount
```

รูปที่ 14 ประโยคที่ใช้อ่านค่าในตัวแ MandalaCount

หน้าต่างบนซ้าย (ใช้คีย์เลื่อนหน้าต่างเลื่อนดู)

แนวความคิด

เพื่อเป็นตัวอย่างอีกเช่นกันและให้เป็นไปตามคู่มือ ST/V ด้วย สมมติว่าเรามีแนวความคิดว่าในการดำเนินการกับอะเรย์ เช่นเป็นอะเรย์ของชื่อลูกค้าที่เรา (สำนักงาน) ติดต่อด้วย โดยอาจนำชื่อไปหาหมายเลขโทรศัพท์หรือจ่าหน้าซองจดหมายก็ได้ อยากทราบว่าในแต่ละวันเราติดต่อถึงใครกี่ครั้ง

นั่นคือเราต้องกำหนดอะเรย์สำหรับนับให้มีจำนวนช่องเท่ากับอะเรย์ตัวจริง ให้ค่าเริ่มต้นเป็นศูนย์ทุกช่อง และเมื่อมีการอ่านค่าจากอะเรย์จริงช่องใดคือ เมื่อใช้ to: ให้เพิ่มหนึ่งที่ช่องนี้ของอะเ นับ

จากงานตามที่กล่าวมานี้เราอ เพิ่มตัวแปรและกิจกรรมในการนับ สกุล Array แต่สำหรับภาษาที่เ โอโอพีมีวิธีการที่ดีกว่าคือ เพิ่มสกุลใ ให้เป็นผู้สืบสกุลของ Array เพราะ ไม่ได้คิดจะให้นับกับอะเรย์ทุกอะเร

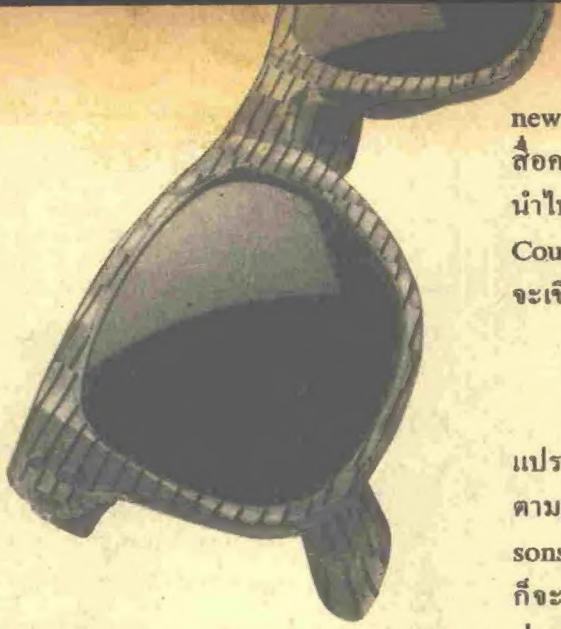
ตัวอย่างโปรแกรมสื่อควา

ผมคิดว่าหากได้เห็นโปรแกรม ความที่จะใช้กิจกรรมในสกุลนี้ก่อน โปรแกรมในรูปที่ 15 คงจะช่วยให้เข้า ว่าเราจะเขียนโปรแกรมกิจกรรมได้อย ไร สำหรับโปรแกรมสื่อความมีร ละเอียดดังนี้

- ได้กำหนดตัวแปรชั่วคราวไ

```
| persons  oneper |
persons := MonitoredArray new: 3.
persons at: 1 put: 'aaa'.
oneper := persons at: 2.
^persons accessCounts
    (0 1 0)
```

รูปที่ 15 ตัวอย่างโปรแกรมสื่อความที่ใช้ MonitoredArray



วคือ persons เพื่อเป็นอะเรย์สำหรับ
ก็บชื่อบุคคล และจะมีอะเรย์แฝด เพื่อ
ก็บจำนวนครั้งที่เราอ่านชื่อในแต่ละ
อง ส่วน oneper จะใช้เก็บชื่อที่อ่าน
ากอะเรย์

- ประโยคในบรรทัดที่ 2 เป็นการ
ำหนดแบบข้อมูลอะเรย์แฝดดังกล่าว
าจำนวน 3 ช่องให้แก่ persons ด้วย
อความ new:

- บรรทัดที่ 3 ให้เก็บชื่อ aaa ที่
องที่ 1 ด้วยสื่อความ at:put:

- บรรทัดที่ 4 ให้อ่านชื่อจากช่อง
2 (ซึ่งขณะนี้ยังว่างอยู่ แต่สมมติว่า
) ให้แก่ oneper ด้วย at: ซึ่งจะต้อง
ำไปดำเนินการต่อ แต่ในที่นี้เพียงแต่ให้
านมาแล้วเก็บไว้เท่านั้น

- บรรทัดสุดท้าย ให้แสดงจำนวน
รั้งที่มีการอ่านกับอะเรย์แต่ละช่อง ดัง
ลการประเมินข้างล่างช่อง 2 มีค่า 1
พราะเราอ่านจากช่องนี้ 1 ครั้ง ส่วน
อง 1 ซึ่งเราเขียนด้วยนั้นคงมีค่าเป็น
นย์ (ขอให้สังเกตว่าสื่อความ at: กับ
t:put เป็นคนละสื่อความกัน)

การกำหนดสกุลใหม่

ก็ต้องขอเรียนให้ทราบก่อนว่า
ณะนี้ผมยังไม่ซึ้งกับภาษาสมอลทอล์ค
อความที่จะกล่าวต่อไปนี้จึงเป็นแต่
พียงที่ผมคิดว่าน่าจะเป็นเท่านั้นครับ

การเขียนโปรแกรมกิจกรรมเมื่อ
ะสร้างสกุลใหม่ จะเริ่มด้วยการกำหนด
ปแบบของสกุลเทียบได้กับการกำหนด
วนหัวของโปรแกรมในภาษาปาสคาล
อจะกำหนดชื่อสกุล และตัวแปรแบบ
าง ๆ

จากแนวความคิดนี้จะได้ว่าเราจะ
ร้างสกุลใหม่โดยให้เป็นลูกของสกุล
rray และจะให้ชื่อสกุลใหม่ว่า Moni-
oredArray ขอให้เลื่อนเคอร์เซอร์ไปที่
น้าต่างบนซ้ายเลือกสกุล Collection
ียกใช้เมนูรองเลือก hide/show เครื่อง
ะแสดงสกุลรองของ Collection ซึ่ง
ะมีสกุล Array อยู่

เลือกสกุล Array เรียกใช้เมนู
รองเลือก add subclass เครื่องจะให้
ป้อนชื่อสกุลรองให้ป้อนเป็น Monitored-
Array เครื่องจะแสดงเมนูมีรายการให้
เลือก 4 รายการซึ่งเป็นแบบของออปเจ็ค
ให้เลือก variableSubclass ไว้ก่อนครับ

เมื่อเลือกรายการนี้แล้วเครื่องจะ
แสดงรูปแบบของสกุลใหม่นี้ในหน้าต่าง
อีดิตดังรูปที่ 16 แต่ตอนนี้ยังไม่มีคำว่า
atCounts (คือนับ at:) ซึ่งเป็นชื่อตัว
แปร (ที่ตั้งขึ้น) เพื่อใช้นับจำนวนครั้ง
ที่มีการอ่านกับอะเรย์แต่ละช่อง ขอให้
เติมคำว่า atCount ให้เป็นดังรูปที่ 16
ครับ

ตัวโปรแกรม

จากโปรแกรมสื่อความที่นำมา
เป็นตัวอย่าง จะเห็นว่ามีการใช้สื่อความ
new:, at:put: และ at: คำพวกนี้เป็น
สื่อความที่เป็นคำเดิม (รายละเอียดจะ
นำไปกล่าวในหัวข้อต่อไป) กับ access-
Counts ซึ่งจะให้ค่าในตัวแปร atCounts
จะเขียนโปรแกรมได้ดังนี้

new:

คำนี้เป็นคำเดิมสำหรับเตรียมค่าตัว
แปรสมาชิกสำหรับเก็บออปเจ็ค เช่น
ตามรูปที่ 15 หากเรากำหนดให้ per-
sons เป็นอะเรย์ธรรมดา new: ของเดิม
ก็จะเป็นผู้เตรียมอะเรย์ให้ตามจำนวน
ช่องที่กำหนด แต่เมื่อเรากำหนดเป็น
อะเรย์แบบใหม่นี้ ใน persons จะมี
อะเรย์ 2 อะเรย์ คืออะเรย์จริงกับอะเรย์
นับ สำหรับอะเรย์นับเรา (ผู้เขียนโปร-
แกรมกิจกรรม) จะต้องเป็นผู้ดำเนินการ
เอง

ภาษาสมอลทอล์ค (เช่นเดียวกับ
ภาษาปาสคาล) เมื่อเรากำหนดคำซ้ำเดิม
จะเท่ากับเป็นการยกเลิกคำเดิม และ
เขียนโปรแกรมใหม่ได้ดังในรูปที่ 17
คำว่า super ที่ใช้คู่กับ new: หมาย
ถึงเรียกใช้ new: ของเดิม เพื่อกำหนด
อะเรย์จริงก่อนแล้วเรียกใช้ initialize
(เป็นโปรแกรมที่เราจะต้องเขียนขึ้นใหม่)
เตรียมอะเรย์นับให้ แล้วส่งค่าใน answer
กลับไป หากเป็นตามตัวอย่างในรูปที่ 15
ก็คือส่งไปให้แก่ persons

ขอให้ป้อนโปรแกรมในรูปที่ 17
เป็นกิจกรรมของสกุลคือ เลือกหน้าต่าง
เล็กเป็น class ตามวิธีการแทรกกิจ-
กรรมดังกล่าวแล้ว เมื่อเซฟแล้วควร
เปลี่ยนหน้าต่างเล็กเป็น instance เพื่อ
กันลืมครับ

อนึ่งโปรแกรมตามรูปที่ 17 และ

```
Array variableSubclass: #MonitoredArray
  instanceVariableNames: 'atCounts'
  classVariableNames: ''
  poolDictionaries: ''
```

รูปที่ 16 ส่วนหัวของสกุล MonitoredArray

รูปที่ 18 จะมีอยู่ในไฟล์ chapter.6 คุณอาจเปิดหน้าต่างใหม่ติดต่อกับดิสก์อ่านข้อมูลในไฟล์นี้มาแล้วใช้copy-paste ข้ามหน้าต่าง มาดำเนินการครั้งละ 1 กิจกรรม

initialize

ใน new: ใหม่ให้ initialize กำหนดอะเรย์นับให้ ดังแสดงในรูปที่ 18 เริ่มด้วยให้ size มีค่าเท่ากับอะเรย์จริง นำมากำหนดให้แก่ atCounts คำว่า new: ในประโยคนี้คือคำเดิมบรรทัดสุดท้ายเป็นการให้ค่าศูนย์แก่แต่ละช่วงในอะเรย์ atCounts ทุกช่อง

at:

ดำเนินการเพิ่มหนึ่งให้แก่อะเรย์ atCounts ช่องที่ anInteger เสร็จแล้วส่งไปให้ at: เดิมดำเนินการต่อ คือให้ค่าที่มีอยู่ในอะเรย์ช่องนี้เช่น ชื่อบุคคล

accessCounts

ให้ค่าในอะเรย์ atCounts ดังผลการประเมินผลในรูปที่ 15

ขอให้แทรกกิจกรรมดังกล่าวมานี้เป็นกิจกรรมสมาชิกครั้งละกิจกรรม (ควรลอกมาจากไฟล์ Chapter.6)

คำเดิม

คำเดิมหมายถึงกิจกรรมที่กำหนดไว้ในบรรพบุรุษ ซึ่งผู้สืบสกุลในตระกูลเดียวกันเท่านั้นที่นำคำเหล่านี้มาใช้ได้ (แต่ในทางกลับกันบรรพบุรุษกลับไม่สามารถใช้คำในผู้สืบสกุล) คำเหล่านี้หากให้กำหนดซ้ำในผู้สืบสกุลจะเป็นการกำหนดทับ หากต้องการใช้คำเดิมให้ใช้คำว่า super นำหน้า สมอลทอล์คจะอยู่ค้นคำในบรรพบุรุษย้อนกลับไปทีละขั้นจนกว่าจะพบ

แต่ก็มีคำส่วนหนึ่งที่กำหนดไว้ในสกุล Behavior เป็นของส่วนรวมให้ทุกสกุลนำไปใช้ได้ เช่น new: เมื่อกำหนดทับหากต้องการใช้คำเดิมให้ใช้ super นำหน้าเช่นกัน ผมจึงกล่าวไว้ในคำเดิม

```
new: anInteger
        "Answer a new MonitoredArray"
    | answer |
    answer := super new: anInteger.
    answer initialize.
    ^answer
```

รูปที่ 17 กิจกรรม new: ที่ต้องเขียนขึ้นใหม่

การเขียนโปรแกรมในแบบโอโอพี

การเขียนโปรแกรมกิจกรรมที่ผมกล่าวมานี้เป็นกิจกรรมเล็กไปหาใหญ่คือเริ่มแทรกจากตัวแปร □ แทรกกิจกรรม และท้ายสุดคือ แทรกสกุล แต่ก็ยังไม่ใช่เป็นลักษณะของการเขียนโปรแกรมในแบบโอโอพี

การเขียนโปรแกรมในแบบโอโอพี (OOP : Object Oriented Progamming) เป็นการเขียนโปรแกรมในเรื่องใดเรื่องหนึ่งโดยเฉพาะ และใช้ประโยชน์ของการสืบสกุลเป็นหลัก ผู้เขียนโปรแกรมควรเป็นผู้เชี่ยวชาญหรือร่วมกันอย่างใกล้ชิดกับผู้เชี่ยวชาญในศาสตร์หนึ่งที่มีลักษณะเป็นการแตกสาขาดังในรูปที่ 19 เช่น ในเรื่องของพืชพันธุ์

เพื่อเป็นตัวอย่างเท่านั้นครับแล ให้ตรงกับ ST/V อีกเช่นกัน ได้ การแยกสกุลของสัตว์ตามรูปที่ 19 เขียนเป็นโปรแกรมในแบบโอโอพี ต รูปนี้แต่ละสกุลแยกเป็น 2 สกุลรอง เพื่อให้น้อยเท่านั้นจะแยกเป็นเท่าไรก็

ใน ST/V ได้แนะนำไว้ว่าในก เขียนโปรแกรมแบบนี้ให้เขียนเป็นโคร ร่างโปรแกรมดังในรูปที่ 20 ก่อน แส ว่าใครสืบสกุลจากใคร และให้กำหน ชื่อตัวแปรไว้ด้วย ซึ่งของจริงจะเป็ ลักษณะเฉพาะของสัตว์ เช่น สี เพื่อเก็บค่า แดง ดำ น้ำตาล เป็นต้น ต ตัวอย่างนี้ก็เพียงแต่กำหนดตามที่คิด ควรจะเป็นเท่านั้น

การที่ให้กำหนดตัวแปรไว้ด้ ก็เพื่อให้เห็นฉนัด เพราะกิจกรรม สืบสกุลสามารถใช้ตัวแปรของบรรพบุรุ

```
accessCounts
        "Answer the array of 'at:' counts."
    ^atCounts


at: anInteger
        "Answer the element in the receiver at
         index position anInteger.  Increment
         the count for accesses to the receiver
         using anInteger."
    atCounts
        at: anInteger
        put: (atCounts at: anInteger) + 1.
    ^super at: anInteger


initialize
        "Private - Initialize the MonitoredArray by
         allocating and initializing the parallel
         atCounts array."
    | size |
    size := self size.
    atCounts := Array new: size.
    1 to: size do: [ :index | atCounts at: index put: 0]
```

รูปที่ 18 กิจกรรมส่วนกลางของสกุล MonitoredArray

ได้ด้วย และโปรดอย่าลืมว่าทั้งตัวแปร, กิจกรรม และสกุลรอง เราสามารถดัดแปลงต่อเติมได้โดยง่าย อันเป็นลักษณะเด่นประการหนึ่งของภาษาที่ให้เขียนโปรแกรมในแบบโอโอพีได้

ดังนั้นตอนนี้จะยังไม่เขียนโปรแกรมทั้งหมด จะเขียนเฉพาะให้สามารถตั้งชื่อสัตว์ได้ และให้สามารถส่งเสียงได้ (talk) และกับสัตว์เพียง 2 ชนิดเท่านั้นก่อนคือ Parrot กับ Dog ซึ่งจะไปเกี่ยวพันกับตัวแปรสมาชิกเพียง 3 ตัวเท่านั้นคือ name, vocabulary และ barksAlot จึงมีโปรแกรมอยู่เพียง 3 ส่วนเท่านั้นคือ

กิจกรรมใน Animal ตามรูปที่ 21 คือ

- name: aString นำค่าคงตัว aString ไปเก็บในตัวแปร name ค่าใน aString คือชื่อสัตว์ที่ผู้เขียนโปรแกรมสื่อความตั้งให้ในฐานะสัตว์เลี้ยง เช่น Lassie (ไม่ใช่ชื่อพันธุ์)

- answer: aString นำข้อความใน aString ไปออกที่บานหน้าต่างหลัก (Transcript Window) โดยมีรูปแบบเป็น

ชื่อสกุล ชื่อสัตว์ : ข้อความ

ตามโปรแกรม ค่าที่ได้จาก self class name คือชื่อสกุล

- talk สำหรับสัตว์ที่ไม่ร้องหรือส่งเสียง เช่น ปลาวาฬ จะให้แสดงข้อความว่า I Can't talk

กิจกรรมใน Parrot ตามรูปที่ 22

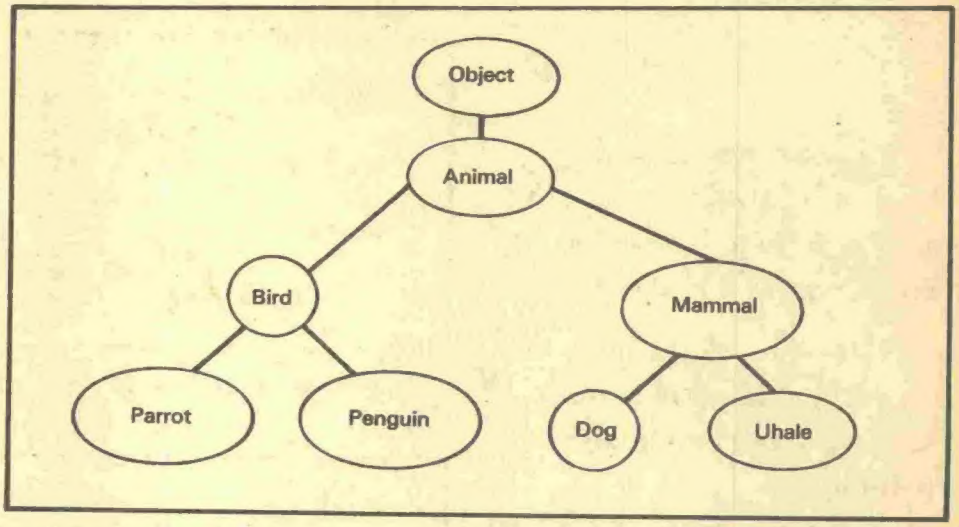
โดยที่เราสามารถฝึกหัดให้นกแก้วพูดเลียนเสียงมนุษย์ได้ การส่งเสียง (talk) ของนกแก้วจึงเป็นประโยคที่ผู้เขียนโปรแกรมสื่อความให้เก็บไว้ใน vocabulary ตามกิจกรรมดังนี้

- Vocabulary: aString นำข้อความใน aString ไปเก็บในตัวแปร vocabulary

ขอให้สังเกตการทำงานของสมอลทอล์คตรงนี้ด้วยครับ โดยพลิกไปดู

- talk การส่งเสียงของนกแก้วจึงเป็นการส่งข้อความในตัวแปร vocabulary ไปให้ Answer นำออกแสดง

โปรแกรมสื่อความในรูปที่ 25 ซึ่งกำหนดตัวแปรโกลบอล Polly ให้เป็นแบบ Parrot และในรูปที่ 26 ประโยค Polly talk นั่นคือ Polly คืออ็อบเจ็คของ talk



รูปที่ 19 แสดงโครงสร้างของสกุล Animal

```
Animal (name, knowledge,habitat,topSpeed, color, picture)
        Bird (flying)
                Parrot (Vocabulary)
                Penguin( )
        Mammal( )
                Dog__(barksAlot)
                Whale( )
```

รูปที่ 20 ร่างโปรแกรมสกุลต่างในสกุล Animal

```
answer: aString
        "Display a message for the receiver animal
         on the System Transcript window, consisting
         of the animal's class name and name preceding
         aString."
    Transcript nextPutAll:
        self class name, ' ', name, ': ', aString;
        cr


name: aString
        "Change the receiver animal's name to aString."
    name := aString


talk
        "Display a message that the receiver can't talk."
    self answer: 'I can''t talk'
```

รูปที่ 21 กิจกรรมใน Animal

เมื่อมาถึงประโยค self answer: voczbulary จึงเป็นการผ่านออปเจ็คทาง self ไปให้แก่ answer หรือกล่าวอีกนัยหนึ่งคือออปเจ็คของ talk ใน Parrot กับออปเจ็คของ answer ใน Animal เป็นออปเจ็คเดียวกัน ทำให้สามารถบ่งสกุลได้ว่าเป็น Parrot

กิจกรรมใน Dog ตามรูปที่ 23

การส่งเสียงของสุนัขคือเห่า ผู้เขียนโปรแกรมนี้ได้แบ่งสุนัขออกเป็น 2 พันธุ์ คือ เห่ามาก กับเห่าน้อย และได้เตรียมตัวแปร barksAlot ไว้ มีกิจกรรมดังนี้

- beNoisy เพื่อให้ barksAlot เป็นจริง และให้แสดงข้อความว่า I will bark a lot (ข้อความนี้ผมเห็นว่าไม่จำเป็น)

- beQuiet เพื่อให้ barksAlot เป็นเท็จ และให้แสดงข้อความว่า I won't bark mush (เห็นว่าไม่จำเป็นเช่นกัน)

- bark เพื่อแสดงเสียงสุนัขเห่าโดยให้ออกเสียงปี๊ด ๆ และแสดงข้อความตามแต่ว่าจะเป็นสุนัขที่เห่ามากหรือเห่าน้อย

- talk แต่เราไม่ได้ให้ผู้เขียนโปรแกรมสื่อความใช้ bark โดยตรง ให้ใช้ talk เหมือนกันหมด ใน talk ของ Dog จึงให้ตรวจว่ามีการให้ค่า barksAlot แล้วหรือยัง ถ้ายังจะใช้ talk ของเดิมใน Animal (คือ I can't talk) ถ้าให้แล้วจึงจะใช้ bark

คุณอาจป้อนโปรแกรมเหล่านี้ได้ด้วยตนเองโดยวิธีการแทรกสกุลและแทรกกิจกรรมดังกล่าวมาแล้ว โดยให้ Animal เป็นผู้สืบสกุลโดยตรงจาก Object แต่ ST/V ได้เตรียมไฟล์เพื่อดำเนินการในเรื่องนี้ไว้แล้ว ขอให้ดำเนินการดังนี้ครับ

- หากเปิดหน้าต่างวงศ์สกุลแล้ว หรือถ้ายังไม่เปิดก็ขอให้เปิดครับจะได้เหมือนกัน ให้เลื่อนกรอบไปอยู่ทางขวาของจอภาพ เพื่อไม่ให้บังหน้าต่างหลักเพราะการดำเนินการต่อไปจะมีการแสดงข้อความที่หน้าต่างหลัก

- เปิดหน้าต่างดิสก์ (การดำเนินการกับดิสก์ : Disk Browser) ทับหน้าต่างวงศ์สกุล โหลดข้อมูลจากไฟล์ chapter.6 มาไว้ในบานอีดิต เลื่อนเคอร์เซอร์ไปประเมินผลข้อความดังในรูปที่ 24

- เรียกหน้าต่างวงศ์สกุลมาอยู่ด้านบน(เช่นใช้ ° cycle ในเมนูหน้าต่าง)

```
vocabulary: aString
        "Change the receiver parrot's vocabulary
         to aString."
    vocabulary := aString


talk
        "Display a message containing the receiver
         parrot's vocabulary."
    self answer: vocabulary
```

รูปที่ 22 กิจกรรมใน Parrot

```
beNoisy
        "Change the status of the receiver dog to noisy."
    barksAlot := true.
    self answer: 'I''ll bark a lot'



beQuiet
        "Change the status of the receiver dog to quiet."
    barksAlot := false.
    self answer: 'I won''t bark much'



bark
        "Have the receiver dog bark by ringing the bell
         and displaying a bark message."
    Terminal bell.
    barksAlot
        ifTrue: [self answer: 'Bow Wow, Bow Wow, Bow Wow!']
        ifFalse: [self answer: 'Woof']



talk
        "Have the receiver dog talk by barking; unless
         barksAlot is nil, in which case the superclass
         can decide how to talk."
    barksAlot isNil
        ifTrue: [super talk]
        ifFalse: [self bark]
```

รูปที่ 23 กิจกรรมใน Dog

```
(File pathName: 'animal6.st') fileIn
```

รูปที่ 24 ประโยคที่อ่านไฟล์ animal6.st ลงหน่วยความจำ

เปิดใช้บานสกุล (บานบนขวา) เรียกใช้เมนูรองเลือกรายการ update

การดำเนินการทั้งหมดนี้เป็นการติดตั้งโปรแกรมดังกล่าวมาเข้ากับระบบเช่นเดียวกันกับที่ป้อนด้วยมือ เพื่อเตรียมการดำเนินการขั้นต่อไป ให้เรียกหน้าต่างดิสค์มาอยู่ด้านบนและเลื่อนเคอร์เซอร์ไปอยู่ในบานอีดิต เพื่อเตรียมประเมินผลข้อความที่จะกล่าวต่อไป ซึ่งได้เตรียมไว้ให้ในบานอีดิตแล้ว

ทดสอบโปรแกรม

เมื่อได้กำหนดสกุลใหม่แล้วจำเป็นอย่างยิ่งที่จะต้องเขียนโปรแกรมสื่อความขึ้นทดสอบ โดยจะแบ่งเป็น 2 ท่อน คือ

- ท่อนแรก

หากเทียบกับภาษาปาสคาลจะเป็นการกำหนดตัวแปรและให้ค่าคงตัว ดังโปรแกรมในรูปที่ 25 ซึ่งได้กำหนดตัวแปร 5 ตัว สำหรับสัตว์ครบทั้ง 4 ชนิดและเพิ่มสุนัขอีก 1 ตัว กับค่าคงตัวต่าง ๆ เช่น ตั้งชื่อสัตว์ ขอให้ประเมินผลประโยคทั้งหมดนี้พร้อมกัน (ด้วย do it)

- ท่อนหลัง

เป็นการใช้งานตัวแปรเทียมได้กับการกำกับการแสดงละครดังโปรแกรมในรูปที่ 26 ขอให้ประเมินผลประโยคเหล่านี้ทีละประโยค (ด้วย do it) เพื่อสังเกตผลที่เกิดขึ้นที่หน้าต่างหลักดังรูปที่ 27

```
Snoopy := Dog new.
Snoopy name: 'Snoopy'.
Snoopy beQuiet.
Lassie := Dog new.
Lassie name: 'Lassie'.
Lassie beNoisy.
Wally := Penguin new.
Wally name: 'Wally'.
Polly := Parrot new.
Polly name: 'Polly'.
Polly vocabulary: 'Polly want a Cracker'.
Moby := Whale new.
Moby name: 'Moby'
```

รูปที่ 25 โปรแกรมสื่อความท่อนแรก

```
"let's hear them talk"

Lassie talk.
Snoopy talk.
Wally talk.
Polly talk; talk; talk.
Polly vocabulary: 'Screeech@#!? Don''t bother me!'.
Polly talk.
Moby talk.
Snoopy beNoisy; talk.
Lassie beQuiet; talk
```

รูปที่ 26 โปรแกรมสื่อความท่อนหลัง

```
Dog Lassie: Bow Wow, Bow Wow, Bow Wow!
Dog Snoopy: Woof
Penguin Wally: I can't talk
Parrot Polly: Polly want a Cracker
Parrot Polly: Polly want a Cracker
Parrot Polly: Polly want a Cracker
Parrot Polly: Screeech@#!? Don't bother me!
Whale Moby: I can't talk
Dog Snoopy: I'll bark a lot
Dog Snoopy: Bow Wow, Bow Wow, Bow Wow!
Dog Lassie: I won't bark much
Dog Lassie: Woof
```

รูปที่ 27 ผลการรันโปรแกรมสื่อความตามรูปที่ 26

ก่อนจบตอน

ผมคิดว่าเรื่องของภาษาสมอลทอล์คที่ผมกล่าวมาตั้ง 3 ตอน คงเพียงพอที่จะทำให้รู้จักรูปร่างภายนอกของภาษาสมอลทอล์คได้ และคงเพียงพอที่จะอ่านโปรแกรมในภาษาสมอลทอล์ค (แบบเดาความด้วย) เพื่อนำมาใช้เขียนในแบบโอโอพี กับโปรแกรมตัวแปลภาษาที่ให้เขียนโปรแกรมในแบบโอโอพีได้ เช่น เทอร์โบปาสคาล 5.5

แต่ถ้าต้องการจะศึกษาเพื่อเลือกว่าหากจะเขียนโปรแกรมในแบบโอโอพีจะใช้ภาษาใดดี ก็ต้องศึกษาสิ่งอำนวยความสะดวกที่สมอลทอล์คโดยเฉพาะที่ใน ST/V มีไว้แล้วในบทที่ 7 ถึงบทที่ 10 ต่อไป

และเมื่อคิดจะนำ ST/V มาใช้จริงก็คงต้องศึกษาเรื่องราวทั้งหมดใน ST/V ที่เหลือ (คืออีกประมาณ 300 กว่าหน้า) และก็ต้องมีหนังสือเรียนภาษาสมอลทอล์คประกอบด้วยครับ

ก็ไม่ใช่เรื่องหนักครับ และสำหรับผมตั้งใจจะนำเสนอถึงบทที่ 10 แต่ขณะนี้ผมจะต้องถนอมสายตาอย่างมากคงจะต้องหาทางเปลี่ยนจอภาพก่อน เพราะ ST/V แสดงตัวหนังสือดำบนพื้นขาว คือเขียว ถึงแม้ผมจะลดความเข้มลงอย่างมากก็ยังรู้สึกเคืองตา